



Angular Fundamentals Module – Observables



Peter Kassenaar
info@kassenaar.com



Hoofdstuk 6
p. 138 en verder



Async services met RxJS/Observables

Reactive programming with asynchronous streams

Async Services

- Statische data ophalen: *synchrone* actie
- Data ophalen uit backend: *asynchrone* actie
- Andere frameworks : `Promises`
- Angular: `Observables`

Bovendien in Angular: ReactiveX library `RxJS`



An API for asynchronous
with observable streams

Choose your platform

<http://reactivex.io/>

Languages

- Java: RxJava
- JavaScript: RxJS
- C#: Rx.NET
- C#(Unity): UniRx
- Scala: RxScala
- Clojure: RxClojure
- C++: RxCpp
- Ruby: Rx.rb
- Python: RxPY
- Groovy: RxGroovy
- JRuby: RxJRuby
- Kotlin: RxKotlin
- Swift: RxSwift

ReactiveX for platforms and frameworks

- RxNetty
- RxAndroid
- RxCocoa

DOCUMENTATION

Observable
Operators
Single
Subject

LANGUAGES

RxJava^ℹ
RxJS^ℹ
Rx.NET^ℹ
RxScala

RESOURCES

Tutorials

COMMUNITY

GitHub^ℹ
Twitter^ℹ
Others

Why Observables?

We can do much more with observables than with promises.

With observables, we have a whole bunch of operators to pull from, which let us customize our streams in nearly any way we want.

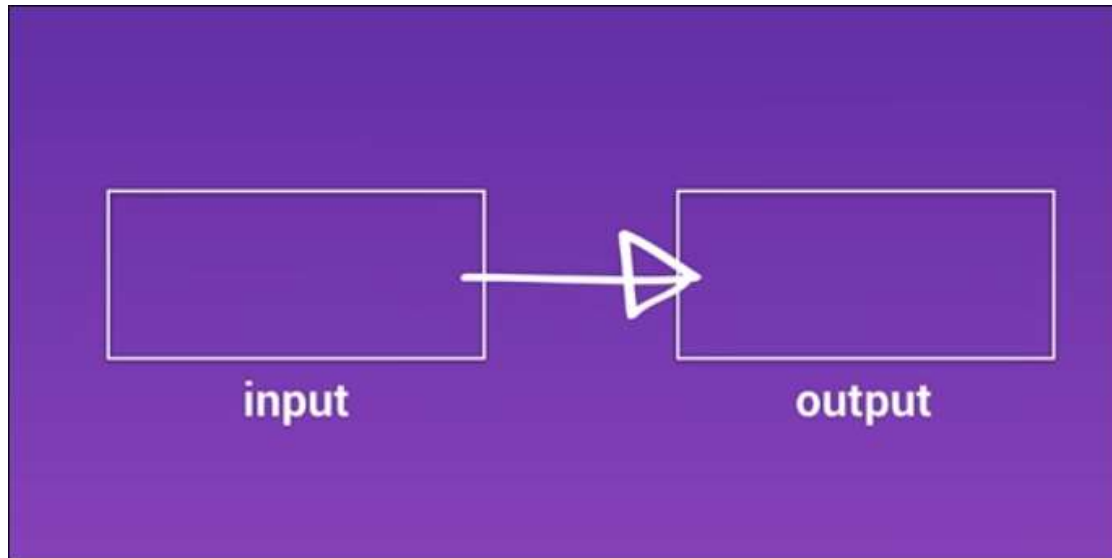
<https://auth0.com/blog/2015/10/15/angular-2-series-part-3-using-http/>

Observables en RxJs

- “Reactive Programming”
 - *“Reactive programming is programming with asynchronous data streams.”*
 - <https://gist.github.com/staltz/868e7e9bc2a7b8c1f754>
- Observables hebben extra mogelijkheden ten opzichte van Promises
 - Mapping
 - Filtering
 - Combining
 - Cancel
 - Retry
 - ...
- Gevolg: géén `.success()`, `.error()` en `.then()` chaining meer!

How do observables work

- First - *The Observable Stream*
- Later - all 10.000 operators...
- Traditionally:





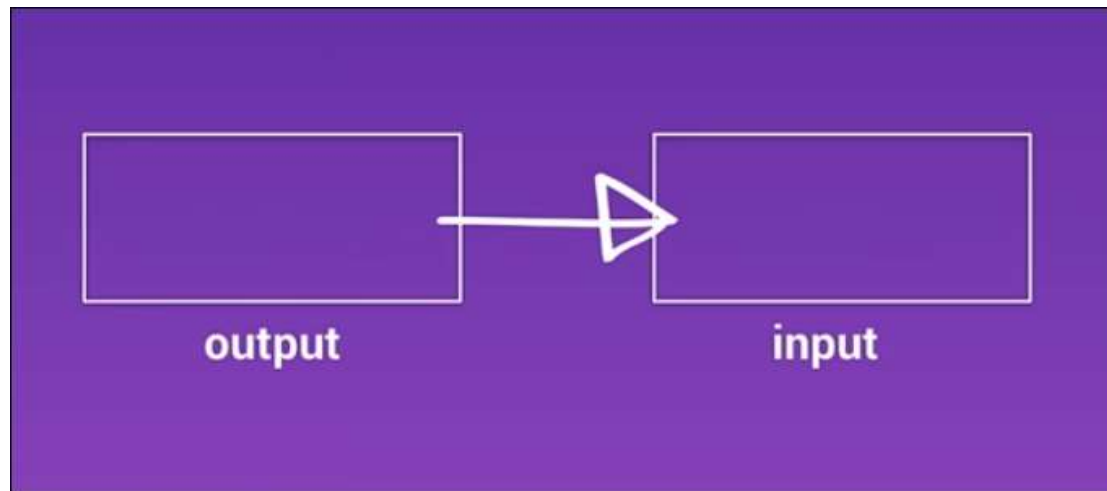
A screenshot of a YouTube video player. The video shows a man standing at a podium with the Angular Connect logo on it. Behind him is a large screen displaying the Angular Connect logo and the text "RANGLE.IO REWRITING THE W". To the right of the screen are logos for Progre, energy, eSynergy, MONACA, Dream UI, and prohire. A subtitle at the bottom of the video reads: "Welcome to go beast mode with realtime interactive interfaces in Angular and Firebase." The video player interface at the bottom shows a play button, a progress bar at 0:06 / 25:21, and icons for settings, full screen, and a list.

Go beast mode with realtime reactive interfaces in Angular 2 & Firebase | Lukas Ruebbelke

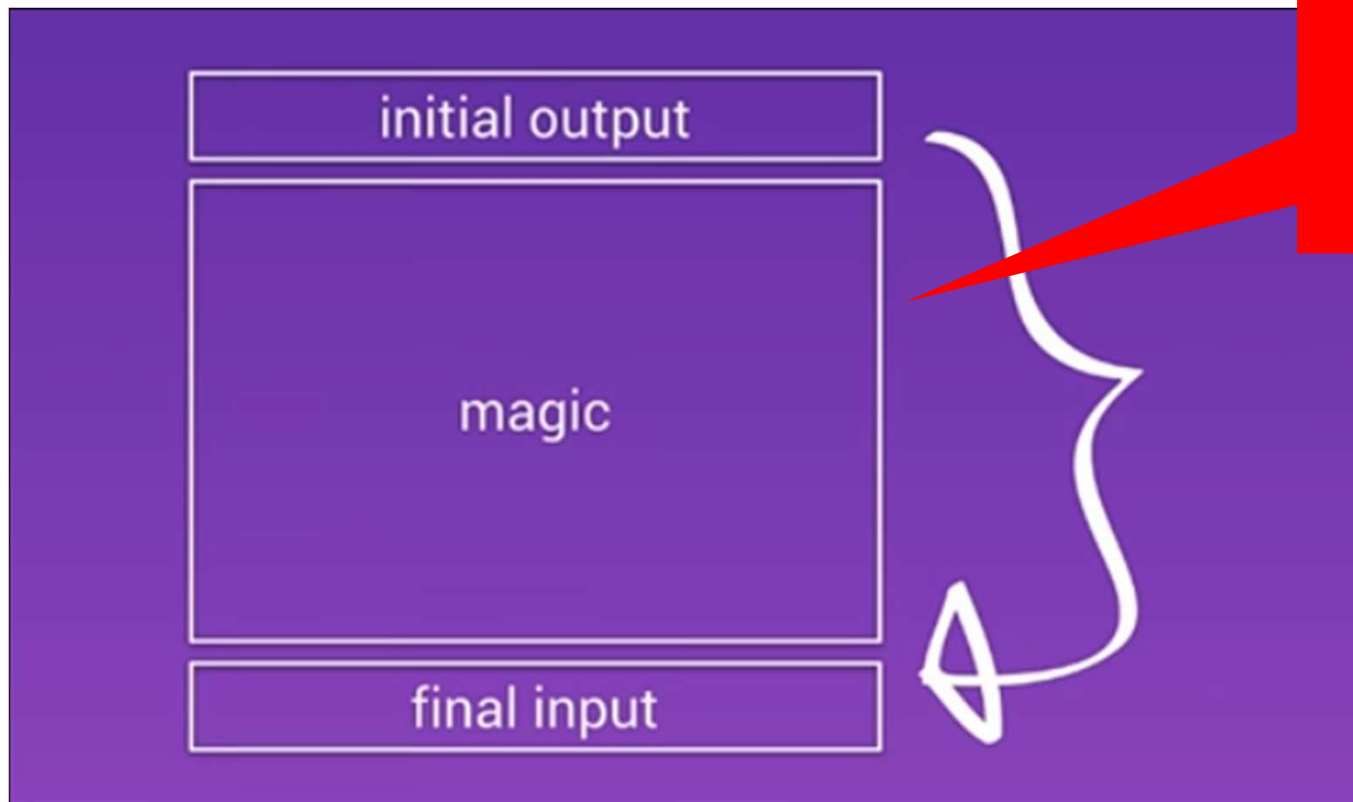
<https://www.youtube.com/watch?v=5CTL7aqSvJU>

<https://youtu.be/5CTL7aqSvJU?t=4m31s>

- With Observables -
 - a system, already outputting data,
 - Subscribe to that data
- "trade Output for Input"
- "Push vs. Pull"

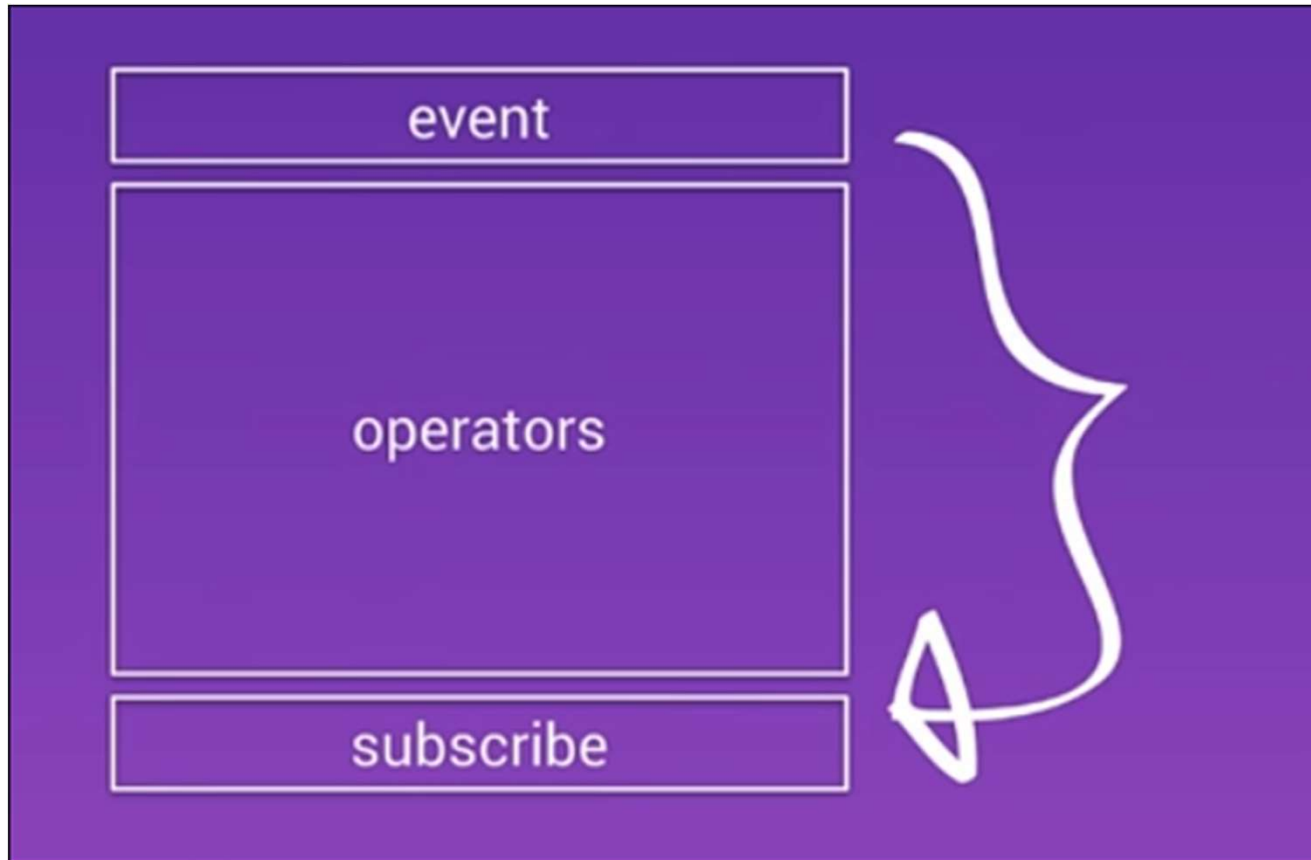


"The observable sandwich"



**Not really Magic.
Just operators**

Subscribe to events



In code:

Initial Output

```
this.http.get<City[]>( 'assets/data/cities.json' )  
    .pipe(  
        delay(...),  
        map(...)  
    )  
    .subscribe((result:City[]) => {  
        //... Do something  
    });
```

Optioneel:
operator(s)

Final Input

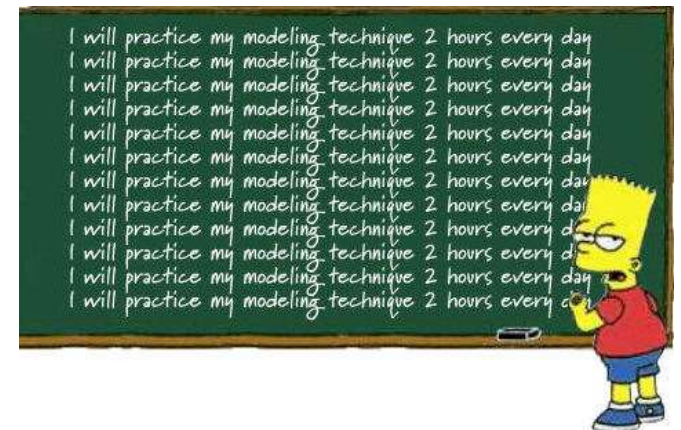
Ook: importeren HttpClientModule in @ngModule

- *// Angular Modules*
...
• **import** {HttpClientModule} **from** '@angular/common/http';
// Module declaration
@NgModule({
 imports : [BrowserModule, HttpClientModule],
 declarations: [AppComponent],
 bootstrap : [AppComponent],
 ...
})
export class AppModule {
}

Oefening

- Bekijk het voorbeeld in `/201_services_http`
- Maak een eigen `.json`-bestand en importeer dit in je applicatie.
- Oefening 5c) , 5d)

Exercise....

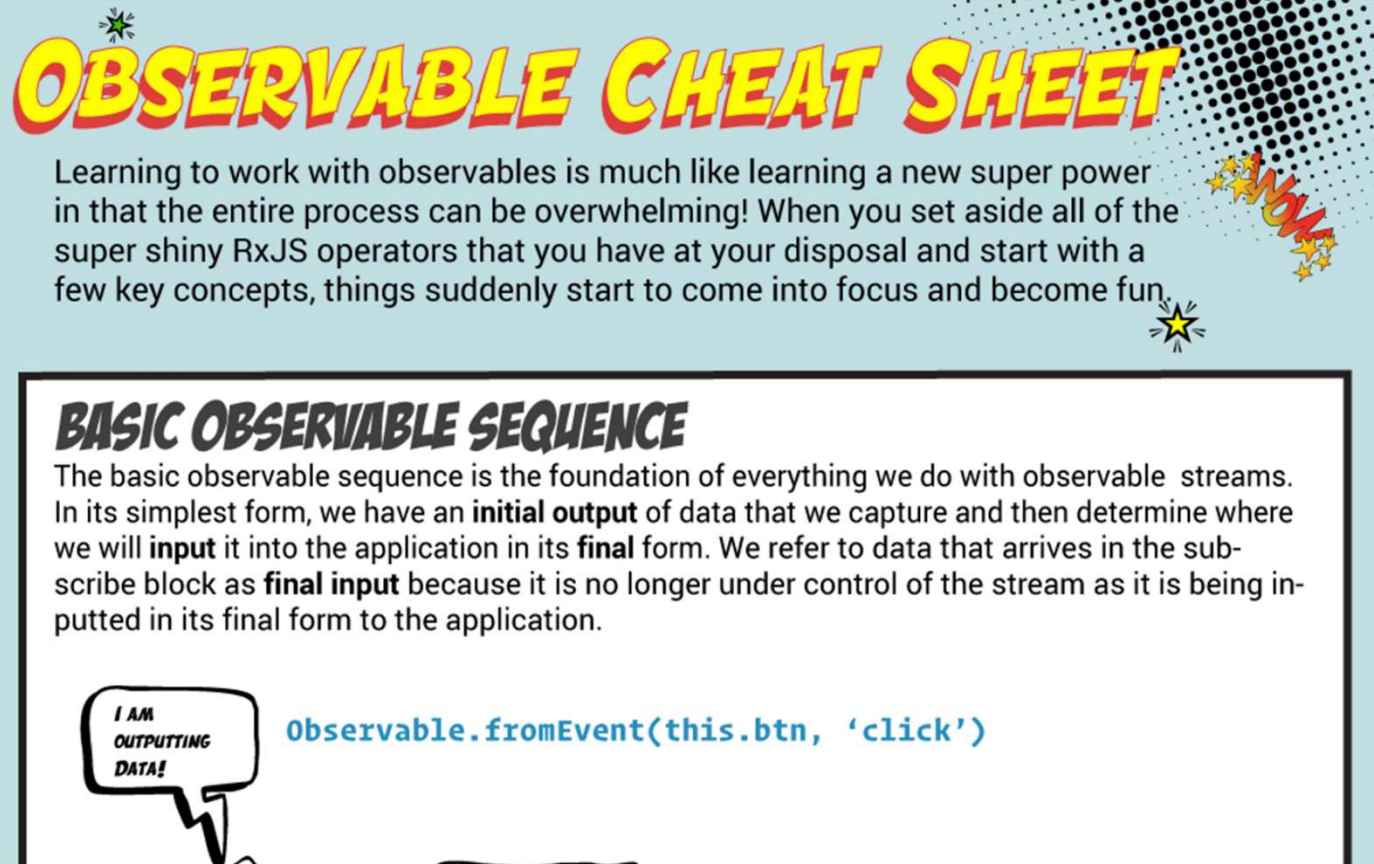


Observable Cheat Sheet

genius to understand.

You can download the full-sized infographic at <http://bit.ly/observable-cheat-sheet>.

I really hope that you find the infographic helpful. Be sure to drop me a line below if you have any questions or comments. #highFive



OBSERVABLE CHEAT SHEET

Learning to work with observables is much like learning a new super power in that the entire process can be overwhelming! When you set aside all of the super shiny RxJS operators that you have at your disposal and start with a few key concepts, things suddenly start to come into focus and become fun.

BASIC OBSERVABLE SEQUENCE

The basic observable sequence is the foundation of everything we do with observable streams. In its simplest form, we have an **initial output** of data that we capture and then determine where we will **input** it into the application in its **final** form. We refer to data that arrives in the subscribe block as **final input** because it is no longer under control of the stream as it is being inputted in its final form to the application.

I AM OUTPUTTING DATA!

```
observable.fromEvent(this.btn, 'click')
```

<http://onehungrymind.com/observable-cheat-sheet/>

Hello RxJS

Gratis online training

The screenshot shows the 'Hello RxJS' micro-course interface. On the left, there's a sidebar with a course box image, a progress bar at 5% complete, and navigation links for 'Class Curriculum' and 'Your Instructor'. The main area is titled 'Class Curriculum' and features a 'Start next lecture' button. Below this is a list of course topics, each with a status icon (circle or document) and a 'Start' button. The topics are: 'Presentation: Realtime Observable Streams' (selected), 'Slides: Realtime Observable Streams', 'The Basic Observable Sequence (2:09)', 'Lab: The Basic Observable Sequence', 'Mapping Values (2:22)', 'Lab: Mapping Values', 'Maintaining State (3:25)' (checked), 'Lab: Maintaining State', 'Merging Streams (1:57)', 'Lab: Merging Streams', 'Mapping to Functions (5:18)', and 'Lab: Mapping to Functions'.

Class Curriculum

Start next lecture > Presentation: Realtime Observable Streams

Hello RxJS

| | | |
|----------------------------------|--|--------------------------------------|
| <input checked="" type="radio"/> | <input type="document"/> Presentation: Realtime Observable Streams | <input type="button" value="Start"/> |
| <input type="radio"/> | <input type="document"/> Slides: Realtime Observable Streams | <input type="button" value="Start"/> |
| <input type="radio"/> | <input type="video"/> The Basic Observable Sequence (2:09) | <input type="button" value="Start"/> |
| <input type="radio"/> | <input type="document"/> Lab: The Basic Observable Sequence | <input type="button" value="Start"/> |
| <input type="radio"/> | <input type="video"/> Mapping Values (2:22) | <input type="button" value="Start"/> |
| <input type="radio"/> | <input type="document"/> Lab: Mapping Values | <input type="button" value="Start"/> |
| <input checked="" type="radio"/> | <input type="video"/> Maintaining State (3:25) | |
| <input type="radio"/> | <input type="document"/> Lab: Maintaining State | <input type="button" value="Start"/> |
| <input type="radio"/> | <input type="video"/> Merging Streams (1:57) | <input type="button" value="Start"/> |
| <input type="radio"/> | <input type="document"/> Lab: Merging Streams | <input type="button" value="Start"/> |
| <input type="radio"/> | <input type="video"/> Mapping to Functions (5:18) | <input type="button" value="Start"/> |
| <input type="radio"/> | <input type="document"/> Lab: Mapping to Functions | <input type="button" value="Start"/> |

<http://courses.ultimateangular.com/>

Pipeable operators

- In RxJS 6.x en hoger: alle operators komen binnen de `.pipe()` functie
- De parameters van de pipe-functie zijn de operatoren!
- Ze worden met komma's van elkaar gescheiden

```
.pipe(  
  delay(3000),  
  retry(3)  
  map(result => ...),  
  takeUntil(...condition...)  
)
```

Subscribe - only once per block!

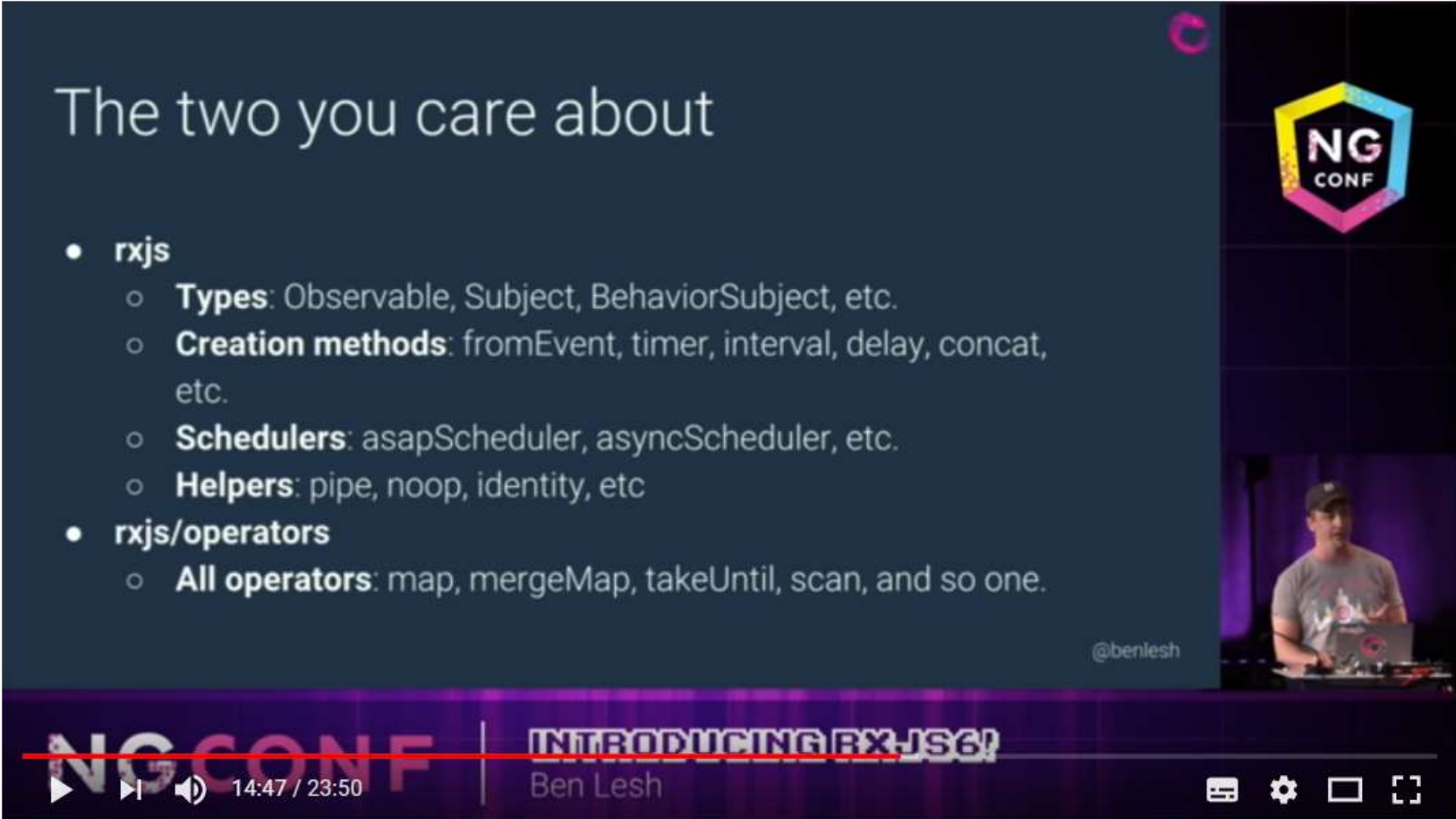
- Three parameters:
 - success()
 - error() – Optioneel!
 - complete() – Optioneel!

```
this.cityService.getCities()
```

```
.subscribe(cityData => {  
    this.cities = cityData;  
},  
err => console.log(err),  
()=> console.log('Getting cities complete...'))
```



Ben Lesh on observables in RxJS 6.0



The video player shows a presentation slide titled "The two you care about" with the following content:

- **rxjs**
 - **Types:** Observable, Subject, BehaviorSubject, etc.
 - **Creation methods:** fromEvent, timer, interval, delay, concat, etc.
 - **Schedulers:** asapScheduler, asyncScheduler, etc.
 - **Helpers:** pipe, noop, identity, etc
- **rxjs/operators**
 - **All operators:** map, mergeMap, takeUntil, scan, and so one.

The video player interface includes the NGCONF logo, the title "INTRODUCING RXJS6!", the speaker's name "Ben Lesh", a progress bar at 14:47 / 23:50, and standard playback controls. A small inset video shows Ben Lesh at a podium.

Introducing RxJS6! - Ben Lesh

<https://www.youtube.com/watch?v=JCXZhe6KsxQ>

Useful operators

- RxJS operators are (mostly) like Array operators
- Perform actions on a stream of objects
- Grouped by subject
 - Creation operators
 - Transforming
 - Filtering
 - Combining
 - Error Handling
 - Conditional and Boolean
 - Mathematical
 - ...

<https://www.learnrxjs.io/>

The screenshot displays the Learn RxJS website. On the left is a sidebar with a search bar labeled 'Type to search'. Below it, the text 'learn-rxjs' is shown. The main navigation menu includes 'LEARN RXJS', 'Introduction' (highlighted in blue), 'Operators', and 'Conditional'. Under 'Operators', a list of specific operators is provided: 'Combination', 'combineAll', 'combineLatest', 'concat', 'concatAll', 'forkJoin', 'merge', 'mergeAll', 'pairwise', 'race', 'startWith', 'withLatestFrom', and 'zip'. The top of the main content area features a header with 'EDIT THIS PAGE', a 'Star' button with a count of 733, and a 'Watch' button with a count of 54. Social media icons for Twitter, Facebook, and GitHub are also present. The main content area has a title 'Learn RxJS' followed by the subtitle 'Clear examples, explanations, and resources for RxJS.' Below this is the 'Introduction' section, which states that RxJS is a popular library for web development, offering a functional approach to dealing with events and integration points. It mentions that learning RxJS is challenging but worthwhile, and that the site aims to provide clear examples and references to help with this. The 'Content' section is partially visible at the bottom.

Type to search

learn-rxjs

LEARN RXJS

[Introduction](#)

Operators

Combination

- combineAll
- combineLatest
- concat
- concatAll
- forkJoin
- merge
- mergeAll
- pairwise
- race
- startWith
- withLatestFrom
- zip

Conditional

EDIT THIS PAGE

Star 733

Watch 54

Learn RxJS

Clear examples, explanations, and resources for RxJS.

Introduction

[RxJS](#) is one of the hottest libraries in web development today. Offering a powerful, functional approach for dealing with events and with integration points into a growing number of frameworks, libraries, and utilities, the case for learning Rx has never been more appealing. Couple this with the ability to utilize your knowledge across [nearly any language](#), having a solid grasp on reactive programming and what it can offer seems like a no-brainer.

But...

Learning RxJS and reactive programming is [hard](#). There's the multitude of concepts, large API surface, and fundamental shift in mindset from an [imperative to declarative style](#). This site focuses on making these concepts approachable, the examples clear and easy to explore, and features references throughout to the best RxJS related material on the web. The goal is to supplement the [official docs](#) and pre-existing learning material while offering a new, fresh perspective to clear any hurdles and tackle the pain points. Learning Rx may be difficult but it is certainly worth the effort!

Content



Async pipe

Automatische `.subscribe()` en `.unsubscribe()`

Async Pipe

- Bij `.subscribe()`, eigenlijk ook `.unsubscribe()` aanroepen.
 - Netjes!
 - Bij HTTP-requests niet beslist nodig, bij andere subscriptions wel, in verband met memory leaks.
- Niet meer zelf `.subscribe()` en `.unsubscribe()` aanroepen:
 - **Gebruik `async pipe` van Angular**

- In de component:

```
Cities$: Observable<City[]>; // Nu: Observable naar Type  
...
```

```
ngOnInit() {  
    // Call naar de service, retourneert Observable  
    this.cities$ = this.cityService.getCities()  
}
```

- In de view:

```
<li *ngFor="let city of cities$ | async">
```

Werken met Live API's

- MovieApp
- `examples\210-services-live`

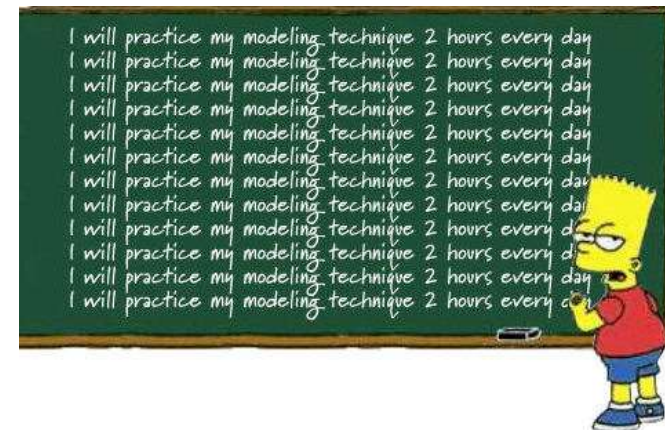


Voorbeeld API's

- <https://pokeapi.co/> - Pokemon API
- <http://openweathermap.org/API> (weerbericht)
- <https://jsonplaceholder.typicode.com/> random users, posts, photos
- <http://ergast.com/mrd/> - Ergast Motor (F1) API
- <http://www.omdbapi.com/> - Open Movie Database
- <http://swapi.co/> - Star Wars API
- Zie ook `JavaScript APIs.txt` met meer voorbeelden

Workshop

- Pick one of your own projects, or see for instance:
 - `../210-services-live`
- Create a small application using one of the API's in the file `JavaScript API's.txt`, using RxJS-calls, for example
 - Pokemon API
 - Kenteken API
 - OpenWeatherMap API
 - ...
- Exercise : 5^e)



Online JSON to TypeScript converter

json2ts

generate TypeScript interfaces from JSON

[email](#) [feedback](#) [help](#)

```
300.jpg"}, {"Title": "The Amazing Captain Nemo", "Year": "1978", "imdbID": "tt0077156", "Type": "movie", "Poster": "https://images-na.ssl-images-  
amazon.com/images/M/MV5BMTc4NzExNjcwN15BMl5BanBnXkFtZTYwMTM1Mjg5_V1_SX300.jpg"},  
{"Title": "Nemo", "Year": "1984", "imdbID": "tt0087784", "Type": "movie", "Poster": "https://images-na.ssl-images-  
amazon.com/images/M/MV5BMTY2NzlwMTgwN15BMl5BanBnXkFtZTcwMjlyMzMzMzMQ@@_V1_SX300.jpg"}, {"Title": "Captain  
Nemo", "Year": "1975", "imdbID": "tt0453375", "Type": "movie", "Poster": "https://images-na.ssl-images-  
amazon.com/images/M/MV5BM2JmOTRlMGQtODMxNy00YmRkLWI1OWEtMmQ2YjZlZmQzZGU5XkEyXkFqcGdeQXVyNDUxNjc5NjY@_V1_  
SX300.jpg"}, {"Title": "Finding Nemo", "Year": "2003", "imdbID": "tt0401422", "Type": "game", "Poster": "N/A"}, {"Title": "Making  
'Nemo'", "Year": "2003", "imdbID": "tt0387373", "Type": "movie", "Poster": "N/A"}, {"Title": "Finding Nemo Submarine  
Voyage", "Year": "2007", "imdbID": "tt1319713", "Type": "movie", "Poster": "https://images-na.ssl-images-  
amazon.com/images/M/MV5BMzAxMzMyODQtNWY0Yy00N2M3LWE5MDQtZDUzNjc1ZGFmMzA4XkEyXkFqcGdeQXVyMzcxMzc4Mw@@_V  
1_SX300.jpg"}, {"Title": "Little Nemo: The Dream  
Master", "Year": "1990", "imdbID": "tt0206895", "Type": "game", "Poster": "N/A"}], "totalResults": "31", "Response": "True"}
```

generate TypeScript

```
declare module namespace {  
  
  export interface Search {  
    Title: string;  
    Year: string;  
    imdbID: string;  
    Type: string;  
    Poster: string;  
  }  
}
```

<http://json2ts.com/>

In VS Code? Use this extension!

EXTENSIONS: MARKET... paste json

Paste JSON as Code 10.0.24
Copy JSON, paste as Go, Type...
quicktype **Install**

JSON Tools 1.0.2
Tools for manipulating JSON
Erik Lynd **Install**

JSON to TS 1.5.5
Convert JSON object to typesc...
MariusAlchimavicius **Install**

Paste Image 1.0.2
paste image from clipboard dir...
mushan **Install**

Paste and Indent 0.0.8
Paste some code with "correct...
g3rry **Install**

Prettify JSON 0.0.3
Visual Studio Code Prettify JS...
Mohsen Azimi **Install**

Sort JSON objects 1.11.0
Sorts the keys within JSON ob...
richie5um2 **Install**

JSON Schema to JSON T.. 1.0.2
Generate json template from js...
ChaunceyKiwi **Install**

json2ts 0.0.6
Convert a JSON from clipboar...
Gregor Biswanger **Install**

Paste JSON as Code quicktype.quicktype
quicktype | 63,602 | ★★★★★ | Repository | License
Copy JSON, paste as Go, TypeScript, C#, C++ and more.
Install

Details Contributions Changelog Dependencies

Visual Studio Marketplace v10.0.24 installs 63602 rating 5/5 (5)

Supports C#, Go, C++, Java, TypeScript, Swift, Elm, and JSON Schema.


quicktype infers types from sample JSON data, then outputs strongly typed models and serializers for working with that data in your desired programming language. For more explanation, read [A first look at quicktype](#).

[Extension Development Host] - pokedex.json

```
{
  "pokedex": {
    "Copy",
    "Copy Line Down",
    "Copy Line Up",
    "Copy With Syntax Highlighting",
    "Developer: Inspect TM Scopes",
    "File: Clear Recently Opened",
    "File: Copy Path of Active File",
    "Poison"
  }
}
```

<https://marketplace.visualstudio.com/items?itemName=quicktype.quicktype>

Data Mocken - Mockaroo

 realistic data generator ? [PRICING](#) [SIGN IN](#)

Need some mock data to test your app?

Mockaroo lets you generate up to 1,000 rows of realistic test data in CSV, JSON, SQL, and Excel formats.

[Need more data? Plans start at just \\$50/year.](#)

| Field Name | Type | Options |
|-----------------------|---------------|----------------------------|
| <div>id</div> | Row Number | blank: 0 % <div>fx</div> × |
| <div>first_name</div> | First Name | blank: 0 % <div>fx</div> × |
| <div>last_name</div> | Last Name | blank: 0 % <div>fx</div> × |
| <div>email</div> | Email Address | blank: 0 % <div>fx</div> × |
| <div>gender</div> | Gender | blank: 0 % <div>fx</div> × |
| <div>ip_address</div> | IP Address v4 | blank: 0 % <div>fx</div> × |

Add another field

Rows:

1000

 Format:

CSV

 Line Ending:

Unix (LF)

 Include: ☒ header ☐ BOM

Download Data

Preview

More

Want to save this for later? [Sign up for free.](#)

<http://mockaroo.com/>

Official documentation...

The screenshot shows the RxJS official documentation website. The left sidebar contains a navigation menu with categories like 'observable', 'observable/dom', 'operator', and 'scheduler'. The main content area is titled 'Observable' and includes a code snippet for importing and using the Observable class. Below the title, it lists 'Direct Subclass' (ConnectableObservable, GroupedObservable, Subject) and 'Indirect Subclass' (AnonymousSubject, AsyncSubject, BehaviorSubject, es6/operator/windowTime.js~CountedSubject, ReplaySubject). A description states: 'A representation of any set of values over any amount of time. This the most basic building block of RxJS.' Under the 'Test:' section, it lists 'Observable', 'Observable.create', and 'Observable.lift'. The 'Static Method Summary' section contains a table with two methods: 'bindCallback' and 'bindNodeCallback'.

```
import {Observable} from '@reactivex/rxjs/es6/Observable.js'
public class | source
```

Observable

Direct Subclass:
ConnectableObservable, GroupedObservable, Subject

Indirect Subclass:
AnonymousSubject, AsyncSubject, BehaviorSubject, es6/operator/windowTime.js~CountedSubject, ReplaySubject

A representation of any set of values over any amount of time. This the most basic building block of RxJS.

Test:
Observable
Observable.create
Observable.lift

Static Method Summary

| Static Public Methods | |
|-----------------------|---|
| public static | bindCallback (func: function, selector: function, scheduler: Scheduler): function(...params: *): Observable Converts a callback API to a function that returns an Observable. |
| public static | bindNodeCallback (func: function, selector: function, scheduler: Scheduler): function(...params: *): Observable |

Generated by ESDoc(0.4.8)

<http://reactivex.io/rxjs/class/es6/Observable.js~Observable.html>

<https://www.learnrxjs.io/>

The screenshot shows the 'Learn RxJS' website. The left sidebar contains a navigation menu with the following items: Introduction, LEARN RXJS (with a sub-menu for Operators), Combination, Conditional, Creation, Error Handling, Multicasting, Filtering, Transformation, Utility, Full Listing, Subjects, Recipes, and Concepts. The main content area is titled 'Operators' and includes a description: 'A complete list of RxJS operators with clear explanations, relevant resources, and executable examples.' Below this is a link: 'Prefer a complete list in alphabetical order?'. The 'Contents (By Operator Type)' section lists three categories: Combination (with sub-items: combineAll, combineLatest, concat, concatAll, endWith, forkJoin, merge, mergeAll, pairwise, race, startWith, withLatestFrom, zip), Conditional (with sub-items: defaultIfEmpty, every, iif, sequenceEqual), and Creation.

Learn RxJS

Introduction

LEARN RXJS

Operators

- Combination
- Conditional
- Creation
- Error Handling
- Multicasting
- Filtering
- Transformation
- Utility
- Full Listing

Subjects

Recipes

Concepts

Powered by GitBook

Operators

A complete list of RxJS operators with clear explanations, relevant resources, and executable examples.

[Prefer a complete list in alphabetical order?](#)

Contents (By Operator Type)

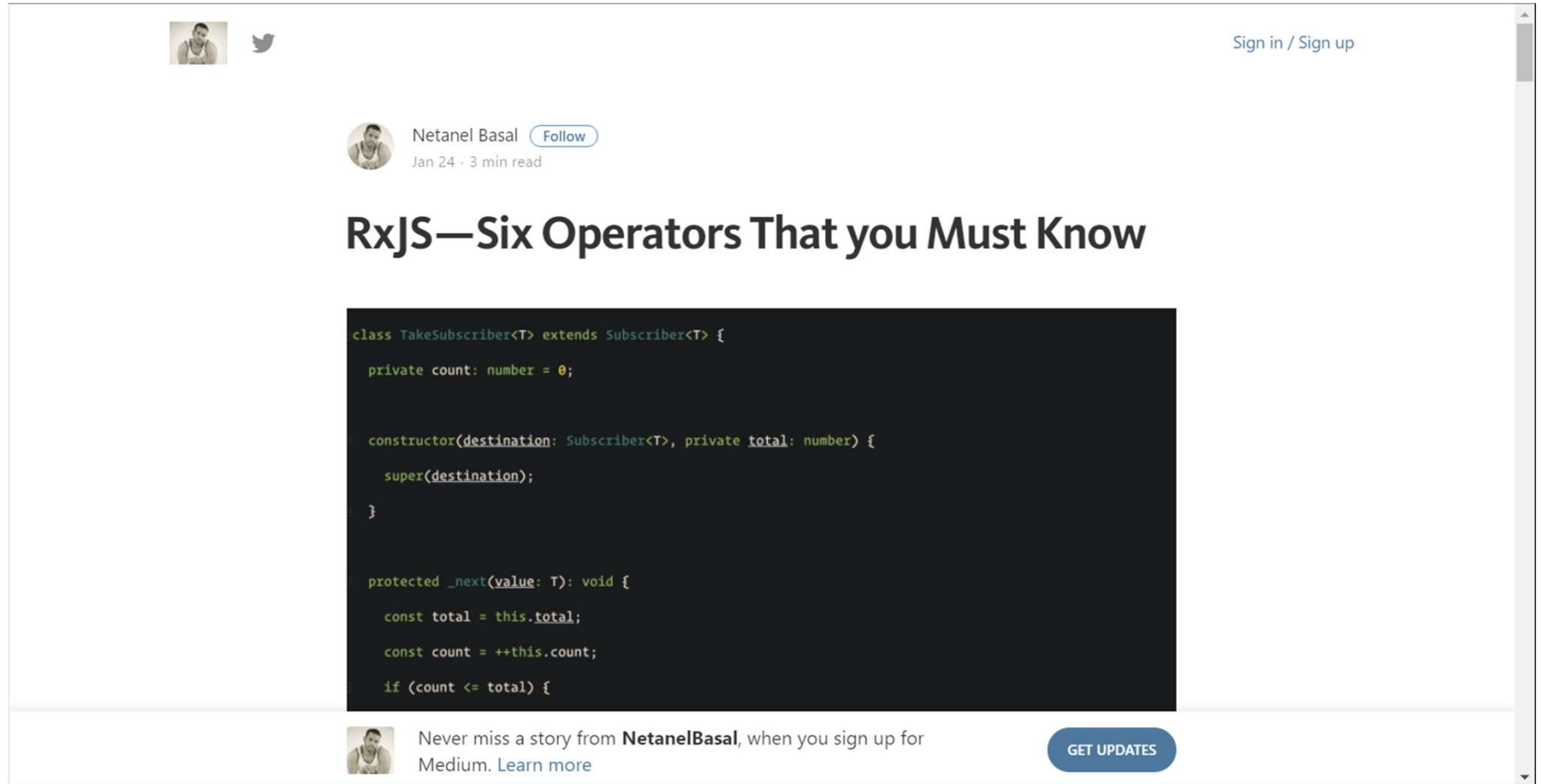
- Combination
 - combineAll
 - combineLatest ★
 - concat ★
 - concatAll
 - endWith
 - forkJoin
 - merge ★
 - mergeAll
 - pairwise
 - race
 - startWith ★
 - withLatestFrom ★
 - zip
- Conditional
 - defaultIfEmpty
 - every
 - iif
 - sequenceEqual
- Creation

CONTENTS

Contents (By Operator Type)

Additional Resources

Article - 6 Operators you must know



The screenshot shows a Medium article preview. At the top left, there's a small profile picture and a Twitter icon. On the top right, it says "Sign in / Sign up". Below this, the author's name "Netanel Basal" is displayed with a "Follow" button and the text "Jan 24 · 3 min read". The article title "RxJS—Six Operators That you Must Know" is prominently displayed. Below the title, a code block is shown with a dark background and light green text. The code is a TypeScript class definition for "TakeSubscriber". At the bottom of the preview, there's a small profile picture, a text prompt "Never miss a story from NetanelBasal, when you sign up for Medium. Learn more", and a blue "GET UPDATES" button.

Sign in / Sign up

Netanel Basal Follow
Jan 24 · 3 min read

RxJS—Six Operators That you Must Know

```
class TakeSubscriber<T> extends Subscriber<T> {  
  private count: number = 0;  
  
  constructor(destination: Subscriber<T>, private total: number) {  
    super(destination);  
  }  
  
  protected _next(value: T): void {  
    const total = this.total;  
    const count = ++this.count;  
    if (count <= total) {
```

Never miss a story from NetanelBasal, when you sign up for Medium. Learn more

GET UPDATES

<https://netbasal.com/rxjs-six-operators-that-you-must-know-5ed3b6e238a0#.11of73aox>

Creating Observables from scratch - André Staltz

The image is a screenshot of a video player. The main content area shows a presentation slide with a blue background. At the top, it says "ng-europe 2016" in a large, light blue font. Below that, there's a photo of a man (André Staltz) speaking at a podium. The podium has a microphone and a laptop. On the podium, there are logos for "JS", "C", and "R". Below the photo, the text "ng-europe.org" is visible. At the bottom of the slide, there's a logo for "RANGLE.IO" with the tagline "SCRIPTING THE WEB". To the right of the presentation slide, there's a code editor window showing JavaScript code. The code defines three callback functions: `nextCallback`, `errorCallback`, and `completeCallback`. It then defines a `giveMeSomeData` function that takes these three callbacks as arguments and calls `document.addEventListener('click', ...)` with an anonymous function that calls `giveMeSomeData`. The code editor has a dark theme and shows line numbers from 1 to 17. The video player interface at the bottom includes a progress bar, a play button, a volume icon, and a timestamp "5:11 / 22:44".

<https://www.youtube.com/watch?v=uQ1zhJHclvs>

The screenshot shows a GitHub Gist page for a file named `introrx.md` by user `staltz`. The page has a header with the GitHub Gist logo, a search bar, and navigation links for 'All gists' and 'GitHub'. The user's profile is shown with a star count of 10,812 and a fork count of 1203. Below the header, there are tabs for 'Code', 'Revisions', 'Stars', and 'Forks'. The 'Code' tab is selected, showing the content of the file. The content is a markdown document titled 'The introduction to Reactive Programming you've been missing' by @andrestaltz. It includes a section 'This tutorial as a series of videos' with a link to 'Egghead.io - Introduction to Reactive Programming'. The document also discusses the challenges of learning Reactive Programming and mentions various libraries like Rx, Bacon.js, and RAC.

GitHub Gist Search... All gists GitHub New gist

staltz / introrx.md Last active an hour ago

★ Star 10,812 Fork 1203

Code Revisions 259 Stars 10812 Forks 1203 Embed <script src="https://gist." Download ZIP

The introduction to Reactive Programming you've been missing

introrx.md Raw

The introduction to Reactive Programming you've been missing

(by @andrestaltz)

This tutorial as a series of videos

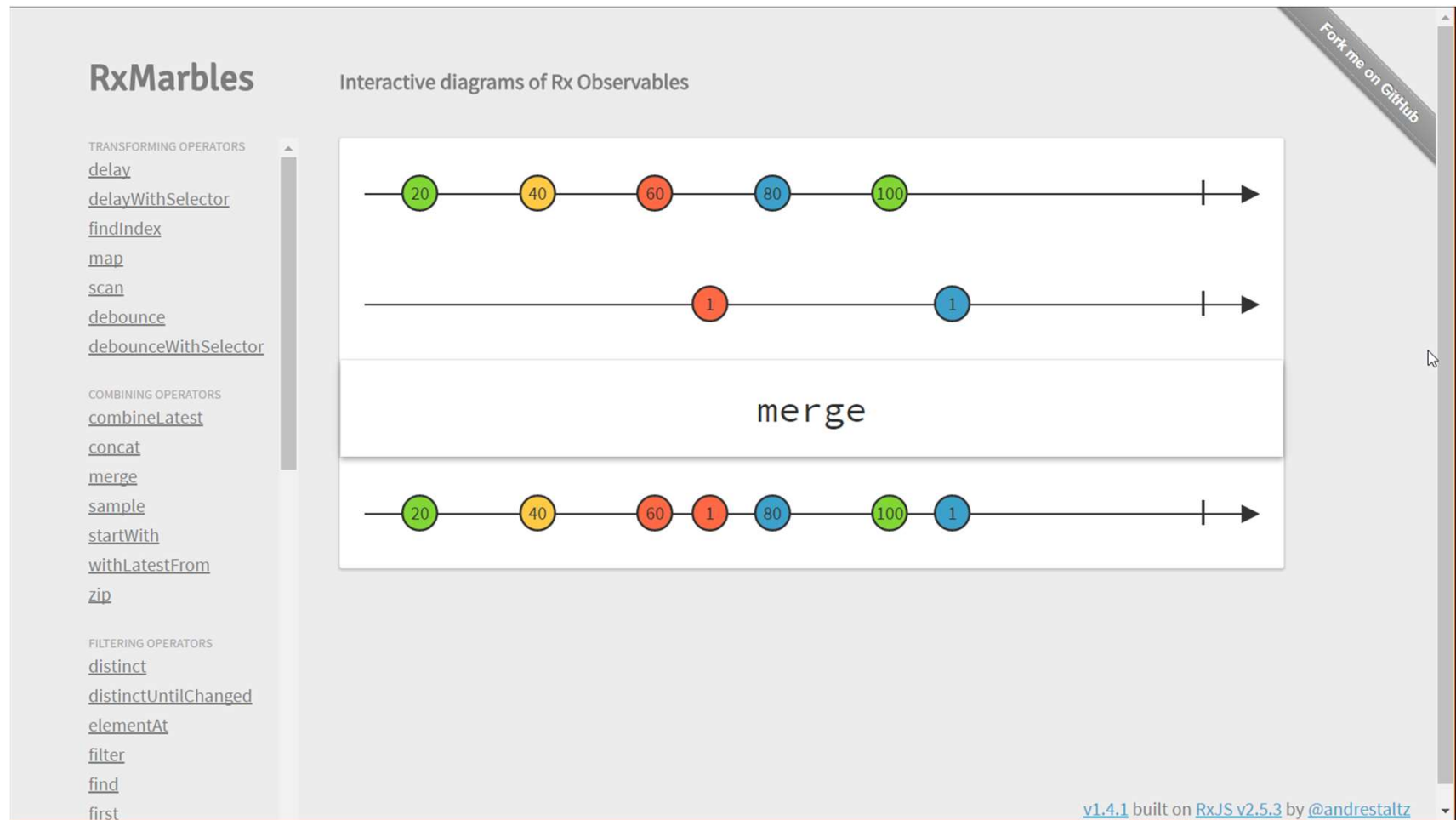
If you prefer to watch video tutorials with live-coding, then check out this series I recorded with the same contents as in this article: [Egghead.io - Introduction to Reactive Programming](#).

So you're curious in learning this new thing called Reactive Programming, particularly its variant comprising of Rx, Bacon.js, RAC, and others.

Learning it is hard, even harder by the lack of good material. When I started, I tried looking for tutorials. I found only a handful of practical guides, but they just scratched the surface and never tackled the challenge of building the whole architecture

<https://gist.github.com/staltz/868e7e9bc2a7b8c1f754>

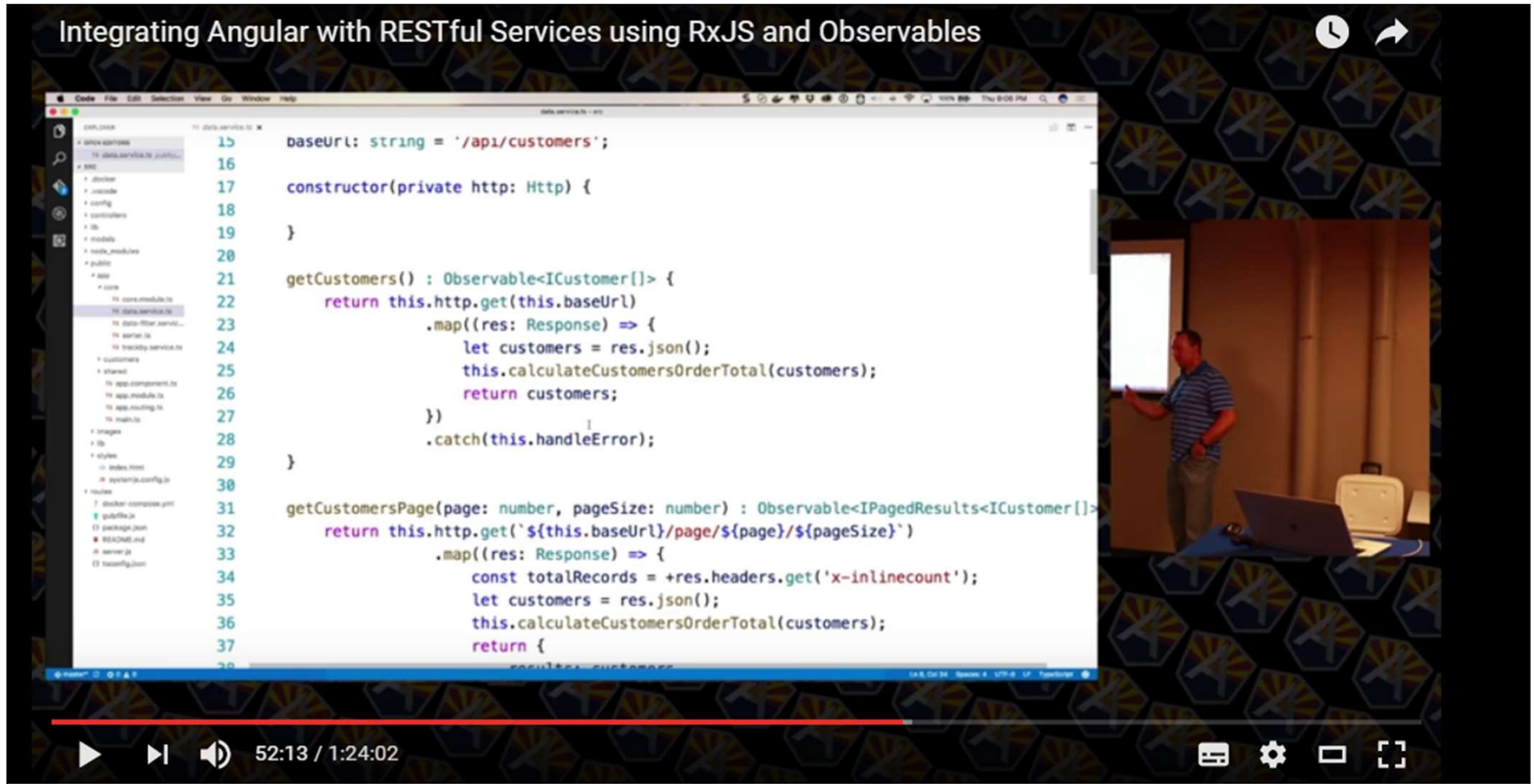
Also by Andre Stalz - RxMarbles



<http://rxmarbles.com/>

Dan Wahlin on Modules and Observables

Integrating Angular with RESTful Services using RxJS and Observables



```
15 baseUrl: string = '/api/customers';
16
17 constructor(private http: Http) {
18 }
19
20
21 getCustomers() : Observable<ICustomer[]> {
22   return this.http.get(this.baseUrl)
23     .map((res: Response) => {
24       let customers = res.json();
25       this.calculateCustomersOrderTotal(customers);
26       return customers;
27     })
28     .catch(this.handleError);
29 }
30
31 getCustomersPage(page: number, pageSize: number) : Observable<IPagedResults<ICustomer[]>
32   return this.http.get(`${this.baseUrl}/page/${page}/${pageSize}`)
33     .map((res: Response) => {
34       const totalRecords = +res.headers.get('x-inlinecount');
35       let customers = res.json();
36       this.calculateCustomersOrderTotal(customers);
37       return {
38         results: customers,
```

52:13 / 1:24:02

<https://www.youtube.com/watch?v=YxK4UW4UfCk>



THOUGHTTRAM

TRAINING

CODE REVIEW

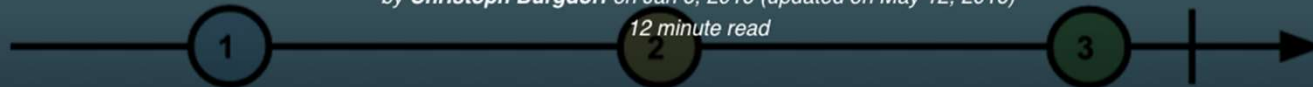
BLOG



TAKING ADVANTAGE OF OBSERVABLES IN `distinctUntilChanged()` ANGULAR 2

by **Christoph Burgdorf** on Jan 6, 2016 (updated on May 12, 2016)

12 minute read



Some people seem to be confused why Angular 2 seems to favor the Observable abstraction over the Promise abstraction when it comes to dealing with async behavior.

There are pretty good resources about the difference between Observables and Promises already out there. I especially like to highlight this free [7 minutes video](#) by [Ben Lesh](#) on [egghead.io](#). Technically there are a couple of obvious differences like the *disposability* and *lazyness* of Observables. In this article we like to focus on some practical advantages that

<http://blog.thoughttram.io/angular/2016/01/06/taking-advantage-of-observables-in-angular2.html>

Een collectie observables ophalen

<https://blog.angularindepth.com/practical-rxjs-in-the-wild-requests-with-concatmap-vs-mergemap-vs-forkjoin-11e5b2efe293>

